Lecture 07

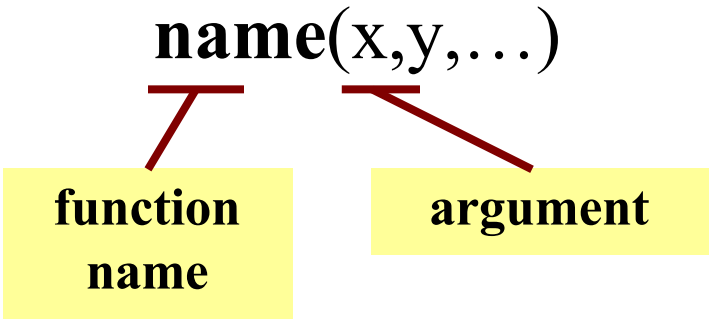
Functions & Modules

Announcements

- No quiz this week
- Assignment 1 will be released this week

Function Calls

- Python supports expressions with math-like functions
  - A function in an expression is a *function call*
- **Function calls** have the form



- **Arguments** are
  - **Expressions**, not values
  - Separated by commas

Built-In Functions

- Python has several math functions
  - round(2.34)
  - max(a+3,24)

Arguments can be any **expression**

- You have seen many functions already
  - Type casting functions: int(), float(), bool()
- Documentation of all of these are online
  - https://docs.python.org/3/library/functions.html
  - Most of these are two advanced for us right now

Functions as Commands/Statements

- Most functions are expressions.
  - You can use them in assignment statements
  - **Example**: x = round(2.34)
- But some functions are **commands**.
  - They instruct Python to do something
  - Help function: help()
  - Quit function: quit()

These take no arguments

- How know which one? Read documentation.

Built-in Functions vs Modules

- The number of built-in functions is small
  - http://docs.python.org/3/library/functions.html
- Missing a lot of functions you would expect
  - **Example**: cos(), sqrt()
- **Module**: file that contains Python code
  - A way for Python to provide optional functions
  - To access a module, the import command
  - Access the functions using module as a *prefix*

Example: Module math

```
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

Example: Module math

```
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

To access math functions

Functions require math prefix!

Example: Module math

```
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

To access math functions

Functions require math prefix!

Module has variables too!

Example: Module math

```
>>> import math
>>> math.cos(0)
1.0
>>> cos(0)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> math.pi
3.141592653589793
>>> math.cos(math.pi)
-1.0
```

To access math functions

Functions require math prefix!

Module has variables too!

Other Modules

- os
  - Information about your OS
  - Cross-platform features
- **random**
  - Generate random numbers
  - Can pick any distribution

Using the from Keyword

```
>>> import math
>>> math.pi
3.141592653589793
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```

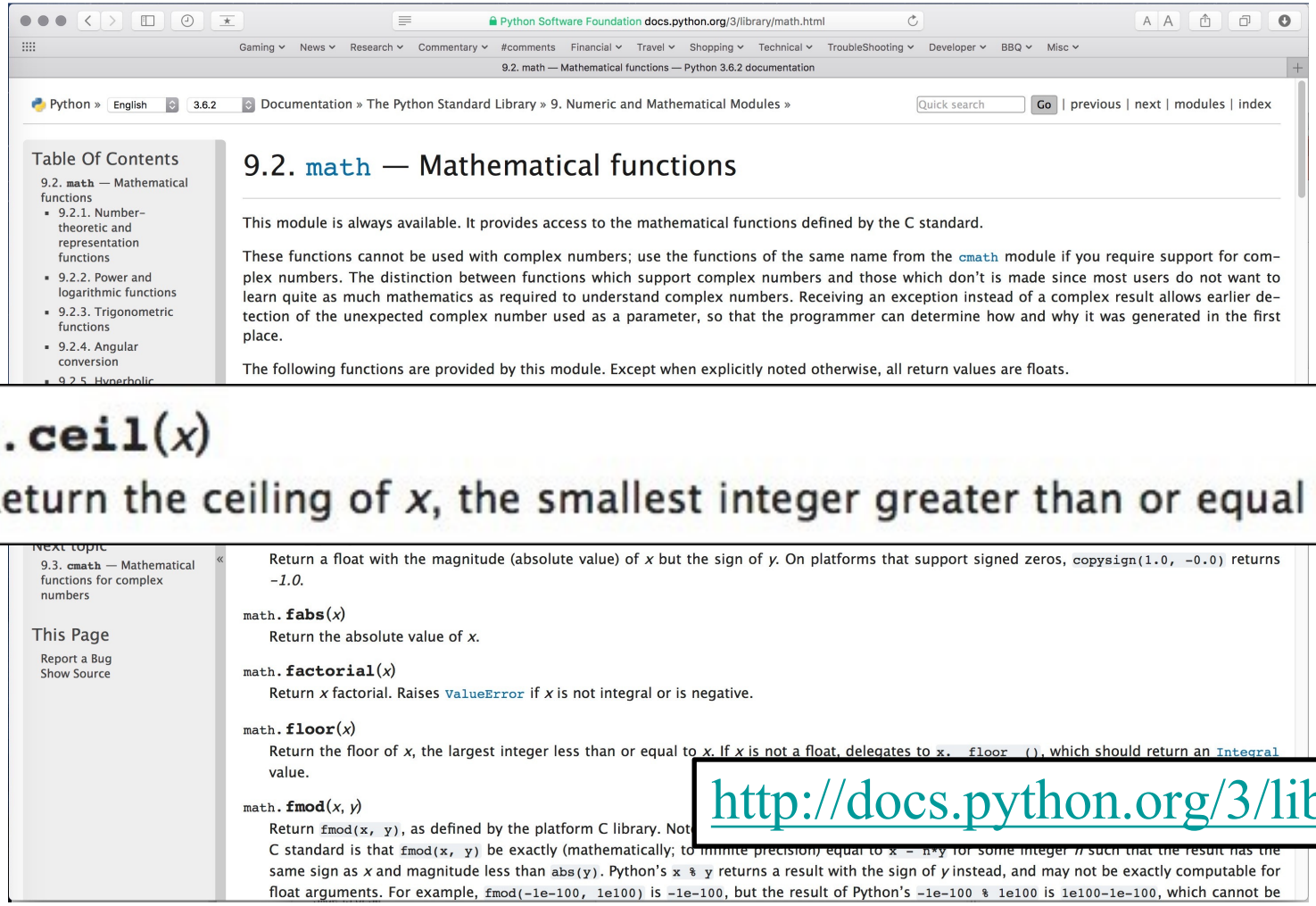
Must prefix with module name

No prefix needed for variable pi

No prefix needed for anything in math

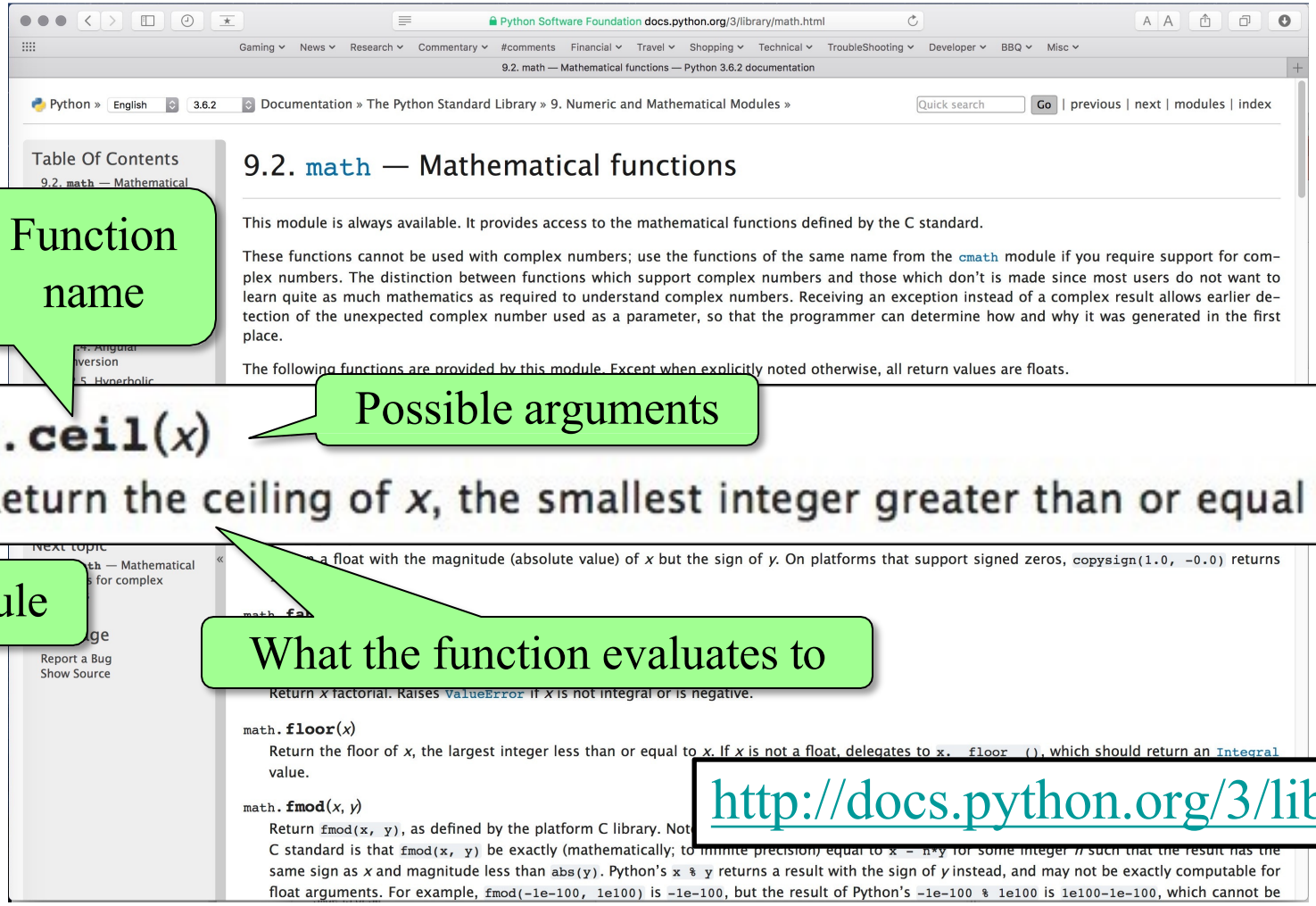
- Be careful using from!
- Using import is *safer*
  - Modules might conflict (functions w/ same name)
  - What if import both?

Reading the Python Documentation



http://docs.python.org/3/library

Reading the Python Documentation



http://docs.python.org/3/library

User Input

```
>>> input('Type something')
Type somethingabc
'abc'
>>> input('Type something: ')
Type something: abc
'abc'
>>> x = input('Type something: ')
Type something: abc
>>> x
'abc'
```

No space after the prompt.

Proper space after prompt.

Assign result to variable.

Numeric Input

- input returns a string
  - Even if looks like int
  - It cannot know better
- You must convert values
  - int(), float(), bool(), etc.
  - Error if cannot convert
- One way to program
  - But it is a *bad* way
  - Cannot be automated

```
>>> x = input('Number: ')
Number: 3
>>> x
'3'
>>> x + 1
TypeError: must be str, not int
>>> x = int(x)
>>> x+1
4
```

Value is a string.

Must convert to int.